Security Considerations in Managing COTS Software

Craig Miller, Cigital, Inc.

2006-12-14

Security failures can have severe consequences whether they are rooted in COTS or custom code. This, coupled with the ubiquity and opacity of COTS software, makes it a critical and difficult problem that an organization ignores at its own extreme peril, however convenient that is to do.

Overview

Information technology tends to focus on new systems—the processes for designing, developing, testing them, and making them secure has been the subject of thousands of books and the focus of hundreds of processes. Building new systems is high-profile, difficult work that receives appropriate attention, but IT operations of an organization rely most heavily on systems that are already in place—the legacy systems. Legacy systems make up the vast bulk of the code base, and all new systems become legacy when they come on line.

Getting security right in new systems is important, but it is equally important, and perhaps even more difficult, to address security in legacy systems. Legacy systems are points of vulnerability in themselves and gateways for attack on new systems. In a modern enterprise, data flows freely between systems, and business processes are instantiated through the orchestration of multiple applications and components. Any point of vulnerability—whether in the physical infrastructure, the underlying operating system and services, new applications, or the oldest legacy application—can compromise the integrity of a process, the data, and the enterprise.

A system may have operated securely for many years, but that is no guarantee that it will continue to do so. The system may simply not have been the target of an attack, the integrity of design may have been compromised by changes made over the years, or a change in an underlying component or service may provide a new avenue of attack.

The issue of changes made to a legacy system is particularly important. By some estimates, 40% to 80% of a system's total lifetime cost is incurred after deployment [Glass 03[1]]. That cost includes upgrades in the underlying system (e.g., new releases of the database), bug fixes, the addition of new features, and changes to accommodate users and other system interfaces. Taken collectively, these changes often rise to the scale of a new system, but much of the work is done outside the new system development process, with substantially less testing and oversight.

The security of legacy systems, then, is as important as that of new systems because they

- are central the organization's operation
- make up the bulk of the code base
- provide a point of entry to the enterprise
- have changed more than most of the organization realizes
- often are changed without the scrutiny given new systems
- are impacted by changes in external systems, services, and infrastructure
- involve old components and technology that may never have been tested

In most organizations, COTS (commercial off-the-shelf) code accounts for the bulk of the legacy environment. The code that the organization develops and maintains in-house is usually the predominant focus of attention. Often the custom code base is prodigious, overshadowing in its impression the COTS elements of the organization's enterprise systems. The attention, however, is often misbalanced. In many

1. #dsy623-BSI_glass03

organizations, the COTS code represents a much larger code base that is more complex and riskier than the internally developed code base.

If you only measure the business-process-specific applications, the body of COTS software may not, in fact, match up to the custom code, but remember that all application software runs on a platform, and that in a modern IT operation, that platform is much more than just the operating system. Platform vulnerabilities are a common point of attack, the methods are widely shared among the hacker community, and the consequences can be severe given the level of access and control than can be obtained if the platform is compromised.

Many organizations trust their COTS code; they assume that it works perfectly and securely. It is convenient and seductive to make this assumption. Worrying about COTS software can keep you up at night and double the workload during the day.

Addressing COTS software security is a very different problem than addressing security in code developed in-house. The problem posed by COTS software is easier in some ways but more difficult in others. In custom code, the organization is the developer, is responsible for all aspects of security, and must work from code out—and most organizations do not really understand secure code development. With COTS code the organization does not have to work with the code; that is the responsibility of the software vendor. Certainly, that is an advantage, since the organization is not called on do something very difficult and likely outside its skill set, but it is also a huge disadvantage. While the organization may not be responsible for code-level security, it does not have an easy way to verify that it was done right, and the organization is the loser if there is a problem.

Why is COTS Software Risky?

COTS Software Presents an Attractive Point of Attack

As security professionals, we tend to focus on the ease with which systems and software security can be compromised. The attack patterns and the tools to implement them are well known to us. When asked about how something can be attacked we say—“Oh, just do A, B, and C (using package D) and you’ll have root,” as if that were simple. It is simple to a professional, but it is not easy in objective terms. The real pros are specialists with considerable knowledge, and the process of breaking code is difficult and time-consuming and has many dead ends. It is hard work.

Hard work demands a high payoff. The payoffs come in the form of intellectual satisfaction and notoriety (even if anonymous) or, more tangibly, damage to the attacked party or access to valuable information. Discovering how to break a COTS package is usually far more attractive than breaking a custom piece of code. Compromises in major packages are well publicized and there is pleasure in “demonstrating” that you are smarter than the security experts at XYZ Corporation. Again, more tangibly, the major COTS packages typically manage important information and connect to more systems. They are central in both the business and technical sense. Further, the information and experience obtained in one attack can be used again on the same package elsewhere.

COTS software is generally a more attractive target than custom code.

COTS Products Are Well Known and Widely Available

The one essential ingredient in attack is access. Access, however, means two different things. Obviously, you must have access to the actual system being attacked. The means for exploring these options are well documented elsewhere [Hoglund 04[2]]. The other aspect of access is the ability to explore the functionality or disfunctionality of the code. This can be done on the system being attacked, but this is not necessarily the case. A version on any system can be used for experimentation. The attacker may be able to use a locally available version, perhaps with legitimate access, to conduct experiments, do reverse engineering, and test attacks. In one case, reviewed recently, the code in question was a thick client. There are literally thousands

2. #dsy623-BSI_hoglund04

of versions available. In some cases, it is possible to download a demo version or purchase an inexpensive, feature- or time-limited version that contains much of the same code.

The other implication of the wide availability of COTS packages is that information is shared among the black hat community. Vulnerabilities and information on viable attack patterns is shared, as is the code to implement the attack.

Microsoft software is the most popular point of attack because of Microsoft's marketplace dominance and the complexity of the products. PowerPoint and Excel in particular have recently become common targets. The ongoing battle between hackers and Microsoft has become so ritualized that the schedule of attacks has fallen into a monthly pattern. For the convenience of systems managers Microsoft has developed a routine of releasing patches on the second Tuesday of the month ("Patch Tuesday"). Hackers have responded by unleashing their attacks in the following days to allow them the maximum time to act unimpeded.

It Is Difficult to Verify the Security of COTS Products

COTS products are generally black boxes to their customers. They can review neither the code nor the architecture. In general, COTS buyers have to rely on the reputation of the developers, published security reports, and security forums. Many software vendors provide public assertions of their security, but these are rarely specific or quantitative. Vendors publish little or nothing about their coding practices as they relate to security. Installation manuals for COTS products are also light on discussions of security.

In theory, black box testing is possible, as will be discussed later, but this can be very difficult for COTS customers. The skills required are outside their experience, but that isn't the principal problem, since they can contract with specialists. The larger challenge lies in all the factors listed previously—the importance of the operating context, the reliance on and interaction with the security infrastructure, and the impact of custom scripting, configuration, and connectivity to external services and data stores. It can be very difficult to test these often very large systems exhaustively in their complex operating context.

COTS Software Vendors Have Very Limited Liability

Among many product categories, there is established legal precedence for product vendors being held liable for the direct and even consequential damages resulting from the failure of their products and sometimes even damages resulting from misuse, particularly when that misuse is not explicitly proscribed. This is not the case with software. Virtually all software comes with a user agreement that explicitly absolves the software vendor of any liability, direct or consequential, even if that failure is a consequence of a known flaw in the product. To use the software, the customer must agree to these terms, and, in general, affected customers have not pursued legal remedies when software has failed. The language below is extracted from an online legal document provided as a model for software product liability. It is consistent with the agreements from several major vendors.

"To the full extent permitted by law, [Vendor] is not liable for any direct, indirect, punitive, incidental, consequential, or exemplary damages (including without limitation loss of business, revenue, profits, goodwill, use of system, or other economic advantage) as a result of or in connection with this software, even if [Vendor] has knowledge of or could reasonably have foreseen the possibility of such damages, however they arise."

The consequences of this sort of limitation and a general business and legal environment that absolves software providers from the liability of other product providers is that a level of failure unequalled in any other product category is accepted and generally deemed the norm. COTS software is a target-rich environment for malicious attack because it is large, complex, and built with fragile and changing technology operating in complex environments that are also constantly changing, and because the liability of its developers is limited by law and by the low expectations of the customers.

Cem Kaner [Kaner 99[3], Kaner 04[4]] has taken on this sort of language, arguing that post-sale warranty disclaimers are invalid and that post-sale limitations on remedies are suspect, having been rejected in several

3. #dsy623-BSI_kaner99

states. He argues that the Uniform Commercial Code (UCC) pertains and offers consumers considerably greater protection.

COTS Software is Generic

COTS software is written without specific knowledge of the enterprise's technical environment or operating procedures. Therefore, COTS code does not typically address instance-specific features of the operating environment. The developer of the COTS code does not know where and how you are going to use it, how you are going to control access, how you will configure the operating system, or anything else specific about your IT operation. So the code will likely lack the specific features necessary to take advantage of your security infrastructure.

How and Why is COTS Software Attacked?

Understanding how and why systems are attacked is central to understanding COTS security issues. You attack different software in different ways if your objective is to simply disable a system versus stealing data. There is no unique way to categorize software attacks. A simple but useful taxonomy defines four categories, listed below. The motivations and means of these attacks differ, so the tools and techniques to deter them vary and the footprints of the attackers also vary.

System Modification Attacks

System modification attacks occur when an attacker takes advantage of a security hole to enter, take over, and modify a target system. These are the most serious forms of attack. Malicious control of a system makes it possible to install viruses, establish a trapdoor, modify account information either to provide illegitimate access or disable legitimate use, replace known and trusted software with a doppelganger, steal confidential information, corrupt data, or install software to monitor and report on usage or forward data on a continuing basis. These are only a few examples of the types of damage that can be done if an attacker obtains control. Lacking the time or skill to do something sophisticated, an attacker may simply use control to render a system inoperable.

COTS systems can provide a point of entry for system modification attacks. The COTS systems of particular interest here are the ones that are externally facing—the email system, the web server, the directory server, the web services infrastructure, etc. Compromising these can potentially provide access to the deepest levels of a system. Virtually no organization creates these general purpose, externally facing systems themselves, and because a small number of products are so widely used, they are extremely attractive to attackers and information on successful attacks is widely shared.

Invasion of Privacy Attacks

Invasion of privacy attacks involve access to information that is meant to be kept private. This includes information such as system and account numbers, although these are generally enciphered or maintained in a directory service, which provides a higher level of security than other systems. The more common theft is application information, particularly financially relevant information. Customer and employee names and other identifying information (social security numbers, employee numbers, etc.) and financial account information are particularly attractive because they are valuable in commerce. These data are very commonly managed by COTS systems. There are some critical differences between privacy attacks and system modification attacks:

- You do not need to obtain complete control of a system to steal data.
- System logs are less likely to show footprints of data theft in privacy attacks than in system modification attacks.
- A system modification attack may be considered successful if the only outcome is damage, while a privacy attack must obtain accurate data and, ideally, conclude without immediate detection.

4. #dsy623-BSI_kaner04

- Corruption of the data may be as valuable as theft.
- COTS applications play a major role in protection against theft of data.

COTS systems typically maintain the data of interest. They serve to collect and concentrate the data in a convenient place. Their security, including access control and possibly enciphering, is critical. All databases and directory servers have their own security models and do not rely exclusively on the operating system security. The interaction of the two is frequently subtle.

Denial-of-Service Attacks

Denial-of-service attacks seek to deny legitimate use of an application, system, computer, or communications link. There are a wide variety of DDoS (distributed denial-of-service) attacks (SYN floods, UDP floods, Teardrop Attacks, etc.), most of which seek to overwhelm a communications connection. Many of these require only a single computer. For example, a SYN flood attack attacks a server by sending multiple requests for connection to a server with forged and nonexistent sender addresses. Because the sender addresses are not real, the server wastes time waiting for a response that never comes. No COTS software is involved, but COTS software is central to application level floods. If COTS software on a server can be corrupted (perhaps through the common buffer overflow), it may be induced to consume system resources to the point of shutting the server down.

COTS software is also central to the initiation of a DDoS attack. DDoS are concurrent attacks on a single system from multiple systems. Essentially, a large number of small, distributed systems all attempt to connect to or invoke a service on the target computer over a short period of time. Typically, these involve the co-option of multiple machines through the distribution of malware. Since multiple machines must be compromised, the only realistic approach is through common software—and that means COTS.

Antagonism Attacks

Antagonism attacks are intended to annoy, harass, or embarrass legitimate users or service providers. They are motivated by ego or malice rather than greed. Generally, they involve the introduction of spurious content (audio or visual) such as obscene, derogatory, or mocking images or an endlessly repetitive sound. These are almost universally delivered over the web and are based on corruption of the material on a site or redirection to content stored elsewhere. The COTS points of attack are the web server and the content management system (CMS) and potentially the database behind the CMS. These must be breached through a hole in the web system security.

Mitigating risk

COTS software is ubiquitous in any organization, so only a comprehensive approach will be effective. Looking at one COTS product out of context has very little meaning. COTS software management must be part of a comprehensive software security risk management. This stands in contrast to the treatment of code developed in-house. Certainly in-house software must be considered in the comprehensive risk-management scheme, and certainly the operating context must be considered in looking at all software, custom or COTS, but a point focus on custom systems is appropriate, particularly when they are new, as the developing organization bears complete and unique responsibility for all aspects of security. There is a further distinction in the approach to mitigation between the COTS and custom systems, rooted in the types of testing and mitigation activities that are appropriate and feasible. Code-level review, rigorous white box testing, etc. are appropriate for in-house code, but neither appropriate nor possible with COTS. The emphasis, then, in COTS testing must be on a systemic approach.

The recommended approach is based on these 16 practices.

1 Identify the COTS Components

You can start in defining COTS by looking at the major packages—databases, web servers, (enterprise resource planning (ERP) packages, etc. The list should be extended to include the smaller and less widely used packages, as these can also compromise the integrity of the enterprise. Just because a package is small does not mean that it is unimportant from a security perspective.

The last section of this paper talks about what COTS means. At first blush, this may seem obvious—COTS is the ERP system or that big package you just wrote a million dollar check for, but COTS is much more than that. In the early days of information technology, programs were monolithic; the boundaries were clear. That is certainly not the case now. Processes at the enterprise levels were once constituted as single programs, but now are frequently instantiated as an orchestration of many independent components; even apparently monolithic blocks of code from a commercial source frequently incorporate code from other sources. It is imperative to recognize the variety and breadth of COTS components.

2 Understand What Counts

Some systems maintain personal, business-private, and critical information; others do not. Some systems are central to the organization's business processes; others are not. Some can provide a point of entry to critical infrastructure; some cannot. Understanding these distinctions is important. You cannot test and protect everything, and it is neither efficacious nor cost-effective to try to do so. You must set priorities. In doing so, it is important to understand the business and legal criticality of the different data and services addressed with COTS. An understanding of the business and regulatory environment is important.

Determining what counts is not a simple task. It inherently requires a dialogue between the IT organization and the business side of the enterprise, two organizations who speak different languages. Security is relative, not absolute. The question "what counts" translates into "what level of risk is acceptable (or optimal)." The answer to that question balances business considerations with technical considerations related to the technical difficulty and cost of mitigation and, paradoxically, the risk introduced by the mitigation.

3 Understand How Things Connect

COTS systems connect with other COTS and custom components both technically (i.e., one calls the other) or in the implementation of a process. Understanding these connections is critical to understanding how the vulnerability in one component may affect other components and how changes in one component can expose (or close) vulnerabilities in others.

The most common point of attack on software is through the input. The interfaces between components are natural points for injecting malicious content to corrupt the data, disable the executable, or alter the behavior of the program.

4 Operate a Secure Computing Infrastructure

The security of COTS software can be compromised if the underlying operating systems, network components, and other elements of the computing infrastructure are not secure. There is little point in building solid systems on mush. The COTS security effort must begin with the environment components. Environment components in this sense certainly include the operating system, but the environment is much more than just the platform. It must also include shared services related to security, message transport, data management, backup, security, and other functions.

5 Control Access

Access control is a fundamental aspect of COTS (and all other) security. COTS systems generally assume that access is controlled in the appropriate way. Access control means electronic systems for firewall and authentication (including tokens and/or biometric means where appropriate). Just as importantly, access control also involves human systems rooted in appropriate division of access and authority, clearly delineated policies and procedures, and training, with frequent reinforcement, monitoring, and audit.

6 Ask the Vendor

This may be too obvious a suggestion, but surprisingly it is often overlooked. The way to ask is not simple yes-no questions, but more specific ones. Ask for a list of security- related problems over the past year or two years. Ask for a list of security-related patches. That problems were identified and fixed is good, but it is irrational to assume, after a long and consistent history of security issues, that the latest one is, finally, the last and that all is now well. Take the frequency of problems as an indication of potential problems, but take the vendor’s diligence in addressing problems as a positive (and appropriate) factor.

7 Engage with the User Community and Security Community

Any significant COTS software package is addressed in at least one online forum. These should be consulted in advance of the purchase decision, as part of the design of the installation, and on a continuing basis during the operations and maintenance phase. Do not assume that all that is said in these forums is accurate, but what is said there should be considered seriously.

8 Engage with Experts

There are myriad security specialists, both individuals and companies. For particularly critical issues, they can provide advice during purchase, assist with the design of the implementation, and, where necessary, assist with testing. Identifying good security consultants, however, is as challenging as identifying secure software. How do you know who is good and whether they have the right capabilities for the specific problem at hand? Checking credentials and reputation is an obvious step, but no firm is right for every engagement. It is important to frame the question well when soliciting assistance. Be precise about what you need, but be open to a creative response. A good security firm will bring its experience to bear on the problem, helping to shape the scope of work, broadening it in some areas, narrowing it in others, and sharpening the focus.

9 Test the Software

The ideal software testing is white box, but this requires access to the source code, which is often not possible with COTS software. No one has found a method to achieve the equivalent level of review working only with the executables. Given this limitation, the best remaining approach is black box testing, in which ranges of data are injected and the results observed. Randomly generated inputs are one approach, but data created specifically to test certain aspects of operation are generally more useful. This approach is practical with smaller applications and entirely sufficient for small software services, but it is impractical for large systems (e.g., ERP systems) because the range of inputs is far too large, particularly when the combinations are considered.

An effective verification method and validation technique for COTS software is fault injection. Faults are injected into a piece of software and the results are observed for anomalous behavior. The value is that that the more tests you perform the more confident you are that the software will behave correctly in use. The challenge is in the design of the faults. This requires an understanding of the software and, in particular, the likely vulnerabilities and modes of failure.

10 Wrap the Software

Fault injections in the testing phase can identify problems, but end users are typically not able to address the problem. Certainly, the defect should be reported to the developer. If there are multiple vulnerabilities with unacceptably adverse consequences, it may be appropriate to consider a different package. If, however, there is no other option or there are compelling reasons to use the package at hand, wrapping the software is an option. Wrapping the software involves interposing another bit of code in front of the application. The front-end code validates the input and only invokes the application if the input is well formed. Wrapping is not simple and does involve risk. A completely effective wrapper must protect against all malformed input. While fault injections can be effective in identifying a particular input or pattern that will compromise the COTS code, it is considerably more difficult to explore completely the range of unacceptable inputs.

One relatively effective method of wrapping is to include the data validation rules of the COTS software in the data validation rules of the database where the inputs to the routine are stored.

11 Look for Certification

Ideally, customers should be able to rely on third-party certification; however, there is no organization that certifies the accuracy, usability, or security of software. ISO (International Organization for Standardization) certification and CMMI (Capability Maturity Model Integration) appraisal speak to the quality of the software vendor and their processes, which is certainly an indicator of software quality, but no organization provides either rigorous standards or testing services that address specific packages. Nonetheless, it is advisable to purchase from vendors who have demonstrated a commitment to quality and to validating this through external assessment.

12 Pay Attention to Updates

Word spreads quickly when a vulnerability is discovered in a popular piece of COTS software. Word may spread in the black hat community if they discover the approach. The security community will generally spread the message to the public and the vendor when they either discover a vulnerability on their own or learn of one discovered by the black hats. When software vendors discover a vulnerability on their own, they generally do not go public until the fix is available. In any case, word spreads quickly and patches follow quickly. It is absolutely critical for the systems management team to be diligent and quick in installing updates.

13 Understand How the COTS Software Operates—To the Extent that You Can

Documentation for COTS software emphasizes what it does, not how it does it. It can be very difficult, therefore, to develop any real understanding of what is going on inside the program, but there are frequently externally visible artifacts. A snapshot of memory can tell you what processes are spawned when a COTS product is running. Some of these are specific to the COTS product, but others are recognizable, and there may be data on their provenance and vulnerability. For example, Zlib is a widely used data compression algorithm. It is used in many applications (and distributed with some operating systems) to speed execution. A flaw was found in late April of 2005 that rendered applications using Zlib vulnerable to a buffer-overflow attack. The problem was fixed quickly, but how can you be sure that the COTS software you are using uses the fixed version? If you see Zlib and cannot determine the version, you should contact the vendor.

Another aspect to watch is the files that are generated by the program. Some may be used as a cache while the programs are running, others cache data between sessions, and others are simply forgotten and left behind. What data do these include: are they encrypted or are they in clear text? The systems teams should examine what is going on before, during, and after execution. Anything in memory or on disk is a potential point of attack.

A commonly used stock-trader client application that can connect to multiple brokerages was found during a recent review to store portfolio information in clear text during a session (encryption was a user option, but not the default) and the files were left behind after execution if the application was aborted rather than terminating normally.

14 Monitor

Major software products typically offer some sort of logging capability. Where logging functionality is not built in, you can certainly add it. These logs should be reviewed periodically for anomalous behavior. One common problem is that the logs are long, and the anomalies are rare. Automated analysis of the logs is almost essential.

We commonly find enterprises that collect vast amounts of data that are ignored or that obfuscate the tiny bit that is irrelevant. Often, the monitoring points are an afterthought, driven by convenience or based on the developers' incomplete understanding of security issues. The design of a monitoring system should be given the same attention as any other new application development—monitoring systems are important. As in all development projects, the design begins with requirements. The first consideration is what data can be collected or what behaviors observed that would indicate that there might be a security problem and to diagnose the problem of one exists. With custom code, there is almost complete flexibility in what can be monitored, but with COTS, the capabilities of the package are constraining.

15 Audit

Periodically, step back and take a fresh look at systems from an objective perspective. This usually requires an outside organization. Their mandate should be broad, but they should be given the information necessary to focus quickly on the points that are critical in either technical or business terms.

16 Prepare in Advance for Failure

No matter how hard an organization works, failure is almost inevitable. Somehow, some way, the integrity of a system may be compromised, a password may be lost, or private data may become public. However diligent an organization is in building for and managing security, they should conduct a “what if” analysis and think hard about the way failure might unfold and how the organization should respond. There should be a technical response to the problem of updating the systems and getting them back online, but there should be a business response as well. The decision of who speaks for the organization and how the organization deals with impacted users should be considered in advance, when there is time and clear heads, rather than at the time of crisis.

How to Think About COTS

COTS products are commercial off-the-shelf software and hardware. These are products that are ready-made and sold as packages. While this paper is specifically focused on software, the discussion here pertains equally to software services instantiated as hardware products—appliances for functions like search or firewall. Similarly, many of the considerations here pertain to MOTS (modifiable off-the-shelf) software. MOTS software is sold like COTS (or downloaded from a community site) but in a form that allows users to modify the code, either through provision of the source code or a scripting language.

The concept of COTS is that of the single-purpose, monolithic program you buy in a shrink-wrapped box. You take it out, load it on your computer (or server) click “run”, and it does what’s advertised—no muss, no fuss, no risk.

The trust is quite misplaced. Few applications are written as monoliths anymore. They are more typically architected as an orchestration of services that may or may not run on a single computer and may or may not be shared with other applications.

COTS Programs Are Customizable

The largest COTS programs are ERP systems. An ERP system such as SAP or Peoplesoft is an immense piece of software. While an ERP system comes with many stock processes representing standard ways of doing things like inventory management and invoicing, every organization is different and, hence, every deployment of the ERP software is different. The difference can be rooted in the idiosyncrasies of an organization’s business processes, the specific sources, structure, and storage of the data, or any of a number of other factors. Whatever the cause, you do not so much buy an ERP as build one using a toolkit. ERP projects are legendary for their complexity and the rate of failure. Often ERP projects do not deliver the desired functionality. So why is it assumed that they deliver the requisite security?

While an ERP package is COTS, a functional ERP system in fact represents a considerable mass of highly complex, business-critical, semi-custom code.

While ERP systems are perhaps the purest example of COTS as a flexible framework rather than a rigid application, this same consideration applies to many less extensive applications. The custom elements may be scripts, database connections, programmatic interfaces, or configuration decisions.

Failure to configure COTS software for secure operation is a particularly common problem. Some common COTS products, such as web servers, are sold for a whole range of applications from simple vanity web pages to high-volume e-commerce sites. Vendors ship the packages with setup scripts that allow the user to set up the software for these very different purposes with very different security considerations. Often, COTS software with the potential for secure operation is compromised when the user selects the wrong configuration options. This is particularly problematic when the setup script is easy to execute, allowing inexperienced and unqualified people to do the installation, and when the insecure option is the default.

The flexibility of COTS packages can cause the same package to be used and operate in fundamentally different ways. COTS does not mean consistent.

Key Observation

Different deployments of a COTS package may be used differently and behave very differently from a security perspective. There are problems in all instances of a package, but it is essential to consider the specifics.

COTS Software Runs in an Environment

COTS software is never the only software operating on a computer. There is an operating system and mechanisms for communication to other systems, typically using IP (internet protocol). Beyond that, there is usually a collection of services on the computer in various forms, which the COTS software uses. Security is not just a function of the COTS program itself but of the COTS program operating as a system with these other components.

Weaknesses in any of these other components can compromise security. Managing security for COTS software means understanding what environment components are used and managing them as well.
Even when nothing has changed in the core COTS software, a change in these components can affect the security of the COTS software, for better or worse. The most pressing areas of concern are the environment components such as web servers that COTS applications use as a gateway to the outside world and other systems. To the extent that the COTS rely on some aspect of these for any aspect of security or validation of the inputs or identity, it is essential that the effect on the COTS applications is considered in evaluating any proposed changes to components and the security impact of significant changes be tested.

The versions of COTS packages for different operating systems are often considerably different. The name of the package is the same, the user experience may be the same, but below the surface, the difference can be considerable. It is virtually impossible for a customer to determine the extent of these differences. For example, is the same code used to validate input strings in the two packages?

Key Observation

The security of a COTS product is highly dependent on the specific environment on which it is installed. Understand the differences between a COTS package running on one operating system versus another. Just because a package is secure on operating system A does not mean that it is secure on operating system B, or just because it is secure on operating system A with one particular revision level and configuration does not mean that it is secure on all installations of that operating system.

COTS Software Operates in a Security Context

COTS software operates in a security context and must be viewed in that context. Few programs operate on computers that are completely isolated from other systems. Accordingly, they must rely on a combination of firewalls, authentication mechanisms, activity monitoring, audits, and policy and procedures for security (hopefully built on an environment of secure code). The COTS software places requirements on the security system (e.g., port access) and is built with specific assumptions about the security infrastructure. Components like the firewall and single sign-on system are of particular concern.

Key Observation

Understand the interaction between the COTS software and the general security context. What requirements does the COTS software impose on the security framework, and how do these requirements affect other systems? How and to what extent does the COTS software rely on external security components? Do systems like single sign-on potentially compromise the security of the COTS software?

COTS Software Is Often a Gateway or Point of Connection

COTS software is important. Nobody writes his own database software, web server, document management system, etc. These major systems are central to an organization in two senses. They are central in the sense that the business cannot operate without them, and they are central in that they connect to many other applications. Every connection to the COTS applications is a potential avenue of attack. How do these other systems authenticate themselves to the COTS application, and how do they send and receive data? Conversely, the COTS software can potentially provide a source of information or point of access to attack the connected systems.

Key Observation

Pay particular attention to COTS software that is used by other packages (e.g., web servers and database systems) and how these connections provide a point of attack on the COTS software and how the connected systems provide an avenue of attack on the COTS software.

COTS Software Is Not All Big Programs

The first things that come to mind when you think of COTS software are the major packages—ERP, database, etc. Much COTS software, however, is less visible. A typical organization has several hundred applications, and larger ones typically have more than a thousand, when you count different versions of the same package. There are always the major systems that command attention and lines in the IT budgets, but the bulk of the COTS software generally lies below the level of attention. These are often packages with a limited number of users or that support functions that are ancillary to the key business activity.

For example, an accounting firm may use a lightweight graphics package for designing presentations or ads. The key accounting packages, the organization’s internal financial systems, and the document management system are more important to the business and contain more personally identifiable and important information, but the small graphics package may access the internet to download upgrades or access a library of clip art or shaders. Hence, they are a point of attack—not as critical as the core systems, but still important.

Another source of COTS vulnerability are add-ins and extensions to major packages. MAYA is the most widely used rendering package, commonly found in organizations doing photorealistic computer graphics. Computer imaging and animation, however, is complex, because the world is complex. MAYA is excellent, but, off-the-shelf, it may not have the specific effect the artist is looking for. Consequently, there are several thousand plug-ins available, for everything from leaves, to fluids, to fire, to smoke. A real “MAYA” installation typically includes a lot more than basic MAYA.

Finally, there are the little bits of COTS code—a custom control here, a DLL (dynamically-linked library) there. These are used by other COTS packages and in code developed in-house. Every COTS installation involves a review of these little components and often involves an upgrade to a newer version. These changes, of course, affect other COTS programs.

In all organizations, the focus is on the security of the major systems that are central to the organization’s business operations and those that contain confidential business and/or customer information. This is appropriate. Failure of these systems has a larger impact, and their visibility makes them an easier and more obvious point of attack. While the major systems are the major problem, they are not the only problem.

Key Observation

COTS software is ubiquitous; it has ramifications beyond the enterprise’s major systems.

A COTS Package Is Not a Process

Organizations do not install software for the sake of having software. They install software to implement and automate a business process. Rarely is the entire process instantiated in a single bit of software. Certainly, the input data must come from somewhere, and that may involve manual or semi-manual processes for data entry, direct data capture from devices, data feeds from remote systems, data obtained from a data store that is populated by other applications, or direct data transfer from other systems (push or pull). The output of the package must go somewhere. It shows up in reports (printed or online), is sent directly to another application (running locally or remotely), is put into a data store to be used by another application, or, perhaps, simply archived. During operations, there may be manual processes or calls to other applications for additional data access, validation of data, specialized calculation, or for many other reasons. Processes are much larger than programs.

A process can be compromised through failure in any component or interception or corruption of the information flow across any of the interstitial connections. When looking at a COTS product, it is essential to think of it in context. It may be very well written code with every security bell and whistle and validated through multiple audits, but if it does not play well with the other constituents of the process, the security is not good enough.

Key Observation

Processes can be compromised even when every component is solid.

So, What Is the Right Way to Think of COTS Software?

When most individuals think of COTS software, they first focus on the major, business-critical packages. Certainly it is important to look at these—they are typically large, complex, and with many points of connection (indicators of risk), they contain and manage important information, and their failure would have a serious adverse impact on the enterprise. In addressing the big packages, it is important to recognize that not all instantiations are the same. Versions differ, there are often different installation options with security implications, and many large packages involve substantial development—scripting, database design, etc.

The world of COTS software, however, does not end with the big packages. The major systems are the starting point, but in the end it is better to adopt a broader definition of COTS—to think of COTS as simply all the bits of code on your system that you did not write and where the source code is not available.

When looking at these bits, it is essential to think of them as living in a context:

- They are built on an environment of operating systems and communications components.
- They operate inside of and interact with a security framework.
- They make use of other small bits of COTS code that are frequently changed.
- Actual processes are much larger then the COTS products used in their construction.

Early application “architectures” consisted of isolated, self-contained applications operating in isolation. Over time, the self-contained applications came to incorporate many off-the-shelf components but were still compiled into a single executable. The ongoing evolution in architecture is towards service-oriented architecture (SOA), in which processes are instantiated as the orchestration of services. Services are nasty things from a security perspective. While they are smaller than applications, they may still be quite complex. They may also run in different memory spaces, communicating with each other through computer buses, local area networks, and the Internet. Each point of communication is a point of attack. The services may also be under the control of an external party who may make changes without consideration of all users’ security or performance requirements and, perhaps, without even informing all users. Change can be frequent and possibly chaotic. The emphasis may shift from COTS applications to COTS components of processes. New methods of addressing security must evolve to address the new architectures.

AJAX (Asynchronous JavaScript and XML) is another emerging technology that introduces some new risk issues. AJAX in intended to create systems with interfaces that are more responsive to user actions by managing the exchange of small bits of data in the background so that it is available on the client computer immediately when called for. The invisible data exchange should receive as much attention from a security perspective as the more visible aspects of the web application design.

Some Summary Observations on Buying and Using COTS Code

COTS software presents a security risk that can be very difficult to measure or control. You cannot look inside COTS software, you have little idea how it is constructed, you do not know what components have been installed, and your ability to test and monitor COTS software is limited. Finally, your COTS software is a visible and attractive target.

But avoiding using COTS software completely is not really an option. The challenge, then, is to buy, deploy, and manage COTS wisely.

A considerable section of this paper is dedicated to a discussion of what COTS software is. This might seem, on first consideration, an exercise in pedantry, but it is not. A subtle, nuanced understanding of COTS is essential. COTS are not just the major packages in your system. COTS range from the platform components (beginning with the operating system), to large applications and small, to bit of code imbedded in and between other components. ***Adopt a broad definition of COTS and learn to recognize it when you see it.***

Buying COTS software is an exercise in research. Most of that research will focus on whether the COTS software will deliver the needed functionality, whether it will work in your environment, and what it costs. Other considerations might address how well it integrates with your other software and whether your organization is able to support it. The security of the software is often not an upfront consideration. The assumption is made that COTS software is secure, until it is proven otherwise. This is not acceptable. ***Consider security as part of the purchase process.***

It is much easier, of course, to say that security should be considered than to do so effectively. Good, objective information from a neutral party is not likely to be available. You must build the security case from multiple sources, each of which is less than adequate. Do online research focusing on security-related sites and user groups. Collect data on past security problems. Consult experts when necessary. Use this background to formulate specific questions for the vendor. ***Research security from all available directions.***

If you have built sufficient confidence to proceed with the purchase, do so cautiously. Run tests. Look at how memory is used. Examine files that are created during execution, and find out how they are handled after normal and abnormal termination. Are any files containing confidential information left behind in an accessible place? What about files containing information that can be used to compromise the software? Perform fault injection tests. ***Assume the software is insecure until proven otherwise.***

Consider the context in which the software operates. Look at the platform components and the components that interface to the COTS packages. How secure are they? Can a vulnerability in one of these compromise the package? Can any deliver malformed input to the COTS package?

Looking at how the systems impacts COTS security is only one part of the problem. The other part is to consider how a security problem with the COTS product will affect other IT systems. ***Consider the environment in which the COTS system will operate; what may be secure in one instance, may not be in another.***

Protect yourself—legally and physically. Request guarantees from COTS vendors and, if possible, indemnification. Where testing or risk indicate, wrap the software with a package that validates input and/or output. ***Protect against failure.***

Security is a consideration during the acquisition and installation of the COTS product and related components. Done right, on deployment, the system is configured in accord with the best practices then extant. As time passes, however, things change. The software may be updated as new attacks develop against the software, an embedded component, or a relevant component of the IT infrastructure. ***Track changes and adapt the security strategy accordingly.***

While planning for change and maintaining the COTS system is critical, sometimes it is not possible to take action in advance of an attack. It is critical, therefore, to put a monitoring system in place to detect failure as soon as possible and to initiate the appropriate action. ***Build in monitoring systems to detect failure.***

Despite best efforts, it is impossible to eliminate the risk of failure completely. Failure may happen. Assume that failure will happen and have a plan in place. This plan should be comprehensive, ranging from repair to the IT systems to addressing the affected parties on behalf of the enterprise. ***Be ready for failure.***

Glossary

security vulnerability	A design flaw or code bug that an attacker could exploit to compromise a system.
security risk	A potential threat to an enterprise represented by the exploitation of a security vulnerability, generally proportional to the likelihood the vulnerability will be exploited and the impact on the enterprise if it is.
COTS	Commercial off-the-shelf software—anything that was bought rather than built.
monitor(ing)	Collecting data on systems in real time, as they are used and updated.
audit(ing)	Conducting a periodic review of systems from basic principles, generally involving an independent team that may be within or external to the organization.
system modification attacks	Attacks designed to alter, corrupt, or disable a system, often to create a gateway for ready malicious access.
invasion of privacy attacks	Attacks designed to obtain access to private information.
denial-of-service attacks	Attacks designed to disable or degrade a system through co-option of critical system resources.
antagonism attacks	Attacks designed to damage the reputation of the attacked organization.

References

[Glass 03]	Glass, Robert L. *Facts and Fallacies of Software Engineering*. Boston, MA: Addison-Wesley, 2003.
[Hoglund 04]	Hoglund, Greg & McGraw, Gary. *Exploiting Software: How to Break Code*. Boston, MA: Addison-Wesley Professional, 2004 (ISBN 0-201-78695-8).
[Kaner 04]	Cem Kaner. "Liability for Defective Content[5]." *ACM SIGDOC'04*, Memphis, TN, October 10–13, 2004.
[Kaner 99]	Kaner, Cem. "A Bad Law for Bad Software—And What We Can Do About It[6]." *Silicon Valley Software Quality Association*, Cupertino, CA, April 13, 1999; also at *Boston Area Software Process Improvement Network*, Boston, April 18, 1999.

[Schneier 06] Schneier, Bruce. "Zero-Day Microsoft PowerPoint Vulnerability[7]." *Schneier on Security* Weblog, July 17, 2006.

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com